



# Software Diversification for WebAssembly

JAVIER CABRERA-ARTEAGA

Doctoral Thesis in Computer Science  
Supervised by  
Benoit Baudry and Martin Monperrus

Stockholm, Sweden, March 2024

TRITA-EECS-AVL-2024:10

ISBN 978-91-8040-822-6

KTH Royal Institute of Technology  
School of Electrical Engineering and Computer Science  
Division of Software and Computer Systems  
SE-10044 Stockholm  
Sweden

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges  
till offentlig granskning för avläggande av Teknologie doktorexamen i elektroteknik  
i .

© Javier Cabrera-Arteaga , March 7th 2024

Tryck: Universitetsservice US AB

## Abstract

WebAssembly, now the fourth officially recognized web language, enables web browsers to port native applications for the Web. Furthermore, WebAssembly has evolved into an essential element for backend scenarios such as cloud computing and edge computing. Therefore, WebAssembly finds use in a plethora of applications, including but not limited to, web browsers, blockchain, and cloud computing. Despite the emphasis on security since its design and specification, WebAssembly remains susceptible to various forms of attacks, including memory corruption and side-channels. Furthermore, WebAssembly has been manipulated to disseminate malware, particularly in cases of browser cryptojacking.

Web page resources, including those containing WebAssembly binaries, are predominantly served from centralized data centers in the modern digital landscape. In conjunction with browser clients, thousands of edge devices operate millions of identical WebAssembly instantiations every second. This phenomenon creates a highly predictable ecosystem, wherein potential attackers can anticipate behavior either in browsers or backend nodes. Such predictability escalates the potential impact of vulnerabilities within these ecosystems, paving the way for high-impact side-channel and memory attacks. For instance, a flaw in a web browser, triggered by a defective WebAssembly program, holds the potential to affect millions of users.

This work aims to harden the security within the WebAssembly ecosystem through the introduction of Software Diversification methods and tools. Software Diversification is a strategy designed to augment the costs of exploiting vulnerabilities by making software less predictable. The predictability within ecosystems can be diminished by automatically generating different, yet functionally equivalent, program variants. These variants strengthen observable properties that are typically used to launch attacks, and in many instances, can completely eliminate such vulnerabilities.

This work introduces three tools: CROW, MEWE, and WASM-MUTATE. Each tool has been specifically designed to tackle a unique facet of Software Diversification. We present empirical evidence demonstrating the potential application of our Software Diversification methods to WebAssembly programs in two distinct ways: Offensive and Defensive Software Diversification. Our research into Offensive Software Diversification in WebAssembly unveils potential paths for enhancing the detection of WebAssembly malware. On the other hand, our experiments in Defensive Software Diversification show that WebAssembly programs can be hardened against side-channel attacks, specifically the Spectre attack.

**Keywords:** WebAssembly, Software Diversification, Side-Channels

## Sammanfattning

WebAssembly, nu det fjärde officiellt erkända webbspråket, gör det möjligt för webbläsare att portera nativa applikationer till webben. Dessutom har WebAssembly utvecklats till en väsentlig komponent för backend-scenarier såsom molntjänster och edge-tjänster. Därmed används WebAssembly i en mängd olika applikationer, däribland webbläsare, blockchain och molntjänster. Trots sitt fokus på säkerhet från dess design till dess specifikation är WebAssembly fortfarande mottagligt för olika former av attacker, såsom minneskorruption och sidokanalattacker. Dessutom har WebAssembly manipulerats för att sprida skadlig programvara, särskilt otillåten cryptobrytning i webbläsare.

Webbsideresurser, inklusive de som innehåller exekverbar WebAssembly, skickas i en modern digital kontext huvudsakligen från centraliserade datacenter. Tusentals edge-enheter, i samarbete med webbläsarklienter, kör miljontals identiska WebAssembly-instantieringar varje sekund. Detta fenomen skapar ett högst förutsägbart ekosystem, där potentiella angripare kan förutse beteenden antingen i webbläsare eller backend-noder. En sådan förutsägbarhet ökar potentialen för sårbarheter inom dessa ekosystem och öppnar dörren för sidkanal- och minnesattacker med stor påverkan. Till exempel kan en brist i en webbläsare, framkallad av ett defekt WebAssembly-program, ha potential att påverka miljontals användare.

Denna avhandling syftar till att stärka säkerheten inom WebAssembly-ekosystemet genom införandet av metoder och verktyg för mjukvarudiversifiering. Mjukvarudiversifiering är en strategi som är utformad för att öka kostnaderna för att exploatera sårbarheter genom att göra programvaran oförutsägbart. Förutsägbarheten inom ekosystem kan minskas genom att automatiskt generera olika programvaruvarianter. Dessa varianter förstärker observerbara egenskaper som vanligtvis används för att starta attacker och kan i många fall helt eliminera sådana sårbarheter.

Detta arbete introducerar tre verktyg: CROW, MEWE och WASM-MUTATE. Varje verktyg har utformats specifikt för att hantera en unik aspekt av mjukvarudiversifiering. Vi presenterar empiriska bevis som visar på potentialen för tillämpning av våra metoder för mjukvarudiversifiering av WebAssembly-program på två distinkta sätt: offensiv och defensiv mjukvarudiversifiering. Vår forskning om offensiv mjukvarudiversifiering i WebAssembly avslöjar potentiella vägar för att förbättra upptäckten av WebAssembly-malware. Å andra sidan visar våra experiment inom defensiv mjukvarudiversifiering att WebAssembly-program kan härdas mot sidokanalattacker, särskilt Spectre-attacken.

# LIST OF PAPERS

1. ***WebAssembly Diversification for Malware Evasion***  
**Javier Cabrera-Arteaga**, Tim Toady, Martin Monperrus, Benoit Baudry  
*Computers & Security, Volume 131, 2023, 17 pages*  
<https://www.sciencedirect.com/science/article/pii/S0167404823002067>
2. ***WASM-MUTATE: Fast and Effective Binary Diversification for WebAssembly***  
**Javier Cabrera-Arteaga**, Nicholas Fitzgerald, Martin Monperrus, Benoit Baudry  
*Submitted to Computers & Security, under revision, 20 pages*  
<https://arxiv.org/pdf/2309.07638.pdf>
3. ***Multi-Variant Execution at the Edge***  
**Javier Cabrera-Arteaga**, Pierre Laperdrix, Martin Monperrus, Benoit Baudry  
*Workshop on Moving Target Defense (MTD 2022), 12 pages*  
<https://dl.acm.org/doi/abs/10.1145/3560828.3564007>
4. ***CROW: Code Diversification for WebAssembly***  
**Javier Cabrera-Arteaga**, Orestis Floros, Oscar Vera-Pérez, Benoit Baudry, Martin Monperrus  
*Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb 2021), 12 pages*  
<https://doi.org/10.14722/madweb.2021.23004>
5. ***Superoptimization of WebAssembly Bytecode***  
**Javier Cabrera-Arteaga**, Shrinish Donde, Jian Gu, Orestis Floros, Lucas Satabin, Benoit Baudry, Martin Monperrus  
*Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming (Programming 2021), MoreVMs, 4 pages*  
<https://doi.org/10.1145/3397537.3397567>
6. ***Scalable Comparison of JavaScript V8 Bytecode Traces***  
**Javier Cabrera-Arteaga**, Martin Monperrus, Benoit Baudry  
*11th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages (SPLASH 2019), 10 pages*  
<https://doi.org/10.1145/3358504.3361228>

# ACKNOWLEDGEMENT

# Contents

<b>List of Papers</b>	<b>iii</b>
<b>Acknowledgement</b>	<b>iv</b>
<b>Contents</b>	<b>1</b>
 <b>I Thesis</b>	 <b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 WebAssembly . . . . .	6
1.2 Predictability in WebAssembly ecosystems . . . . .	8
1.3 Problem statements . . . . .	9
1.4 Approach: Software Diversification . . . . .	9
1.5 Summary of research papers . . . . .	10
1.6 Thesis outline . . . . .	12
 <b>2 Background and state of the art</b>	 <b>13</b>
2.1 WebAssembly . . . . .	13
2.1.1 From source code to WebAssembly . . . . .	14
2.1.2 WebAssembly’s binary format . . . . .	17
2.1.3 WebAssembly’s runtime . . . . .	18
2.1.4 WebAssembly’s control-flow . . . . .	20
2.1.5 Security and reliability for WebAssembly . . . . .	21
2.1.6 Open challenges . . . . .	22
2.2 Software diversification . . . . .	23
2.2.1 Automatic generation of software variants . . . . .	23
2.2.2 Equivalence Checking . . . . .	26
2.2.3 Variants deployment . . . . .	27

2.2.4	Measuring Software Diversification . . . . .	28
2.2.5	Offensive or Defensive assessment of diversification . . . . .	29
2.3	Open challenges for Software Diversification . . . . .	30
<b>3</b>	<b>Automatic Software Diversification for WebAssembly</b>	<b>32</b>
3.1	CROW: Code Randomization of WebAssembly . . . . .	33
3.1.1	Enumerative synthesis . . . . .	33
3.1.2	Constant inferring . . . . .	35
3.1.3	Exemplifying CROW . . . . .	36
3.2	MEWE: Multi-variant Execution for WebAssembly . . . . .	38
3.2.1	Multivariant call graph. . . . .	39
3.2.2	Exemplifying a Multivariant binary . . . . .	40
3.3	WASM-MUTATE: Fast and Effective Binary Diversification for WebAssembly . . . . .	41
3.3.1	WebAssembly Rewriting Rules . . . . .	42
3.3.2	E-Graph traversal . . . . .	44
3.3.3	Exemplifying WASM-MUTATE . . . . .	44
3.4	Comparing CROW, MEWE, and WASM-MUTATE . . . . .	47
3.4.1	Security applications . . . . .	50
<b>4</b>	<b>Assessing Software Diversification for WebAssembly</b>	<b>52</b>
4.1	Offensive Diversification: Malware evasion. . . . .	52
4.1.1	Cryptojacking defense evasion . . . . .	53
4.1.2	Methodology . . . . .	54
4.1.3	Results . . . . .	56
4.2	Defensive Diversification: speculative side-channel protection . . . . .	59
4.2.1	Threat model: speculative side-channel attacks . . . . .	60
4.2.2	Methodology . . . . .	61
4.2.3	Results . . . . .	63
4.3	Conclusions. . . . .	67
<b>5</b>	<b>Conclusions and Future Work</b>	<b>68</b>
5.1	Summary of technical contributions . . . . .	68
5.2	Key results of the thesis . . . . .	69
5.3	Future Work . . . . .	70
5.3.1	Data augmentation for Machine Learning on WebAssembly programs. . . . .	70



5.3.2	Improving WebAssembly malware detection via canonicalization . . . . .	71
5.3.3	Oneshot Diversification . . . . .	72

<b>References</b>	<b>73</b>
-------------------	-----------

<b>II Included papers</b>	<b>89</b>
---------------------------	-----------

WebAssembly Diversification for Malware Evasion	<b>91</b>
-------------------------------------------------	-----------

WASM-MUTATE: Fast and Effective Binary Diversification for WebAssembly	<b>92</b>
---------------------------------------------------------------------------	-----------

CROW: Code Diversification for WebAssembly	<b>93</b>
--------------------------------------------	-----------

Multi-Variant Execution at the Edge	<b>94</b>
-------------------------------------	-----------

Superoptimization of WebAssembly Bytecode	<b>95</b>
-------------------------------------------	-----------

Scalable Comparison of JavaScript V8 Bytecode Traces	<b>96</b>
------------------------------------------------------	-----------

**Part I**

**Thesis**